1 - Introduction

Notion d’algorithme

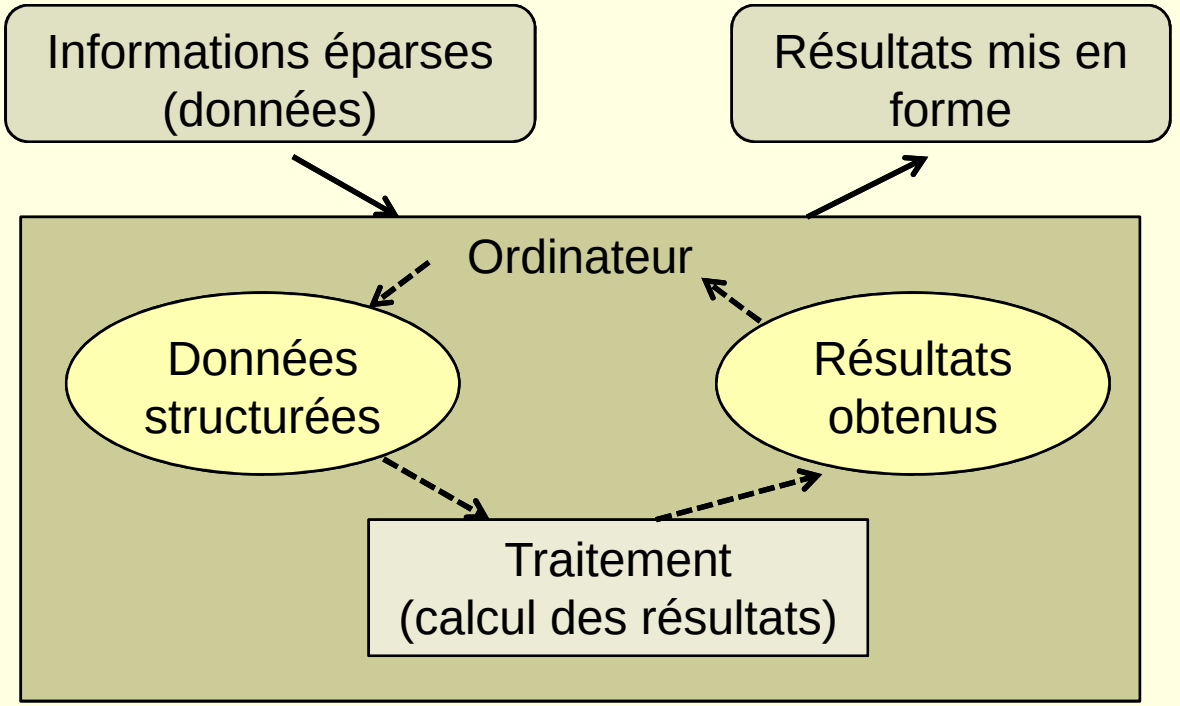
1.1 – Définitions

- **Informatique** : domaine concernant le traitement de l'information à l'aide d'un ordinateur

- **Ordinateur** : dispositif permettant de manipuler des **informations**, les ***données***, en vue d'obtenir de nouvelles informations, les ***résultats***.

Pourquoi un cours d'Algorithmique ?

- Objectif: obtenir de l'ordinateur qu'il effectue un travail à notre place
- Problème: expliquer à l'ordinateur comment il doit s'y prendre



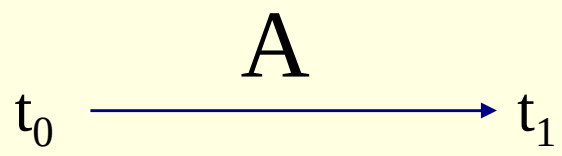
1.1 – Définitions : action, indicateur

- **Action** : événement produit par un acteur, l'exécutant. Elle prend place pendant une période finie et produit un résultat bien défini et prévu.
- **Indicateur** : un indicateur est caractérisé par son nom et sa valeur, il représente une information.

Une action agit sur un ensemble d'indicateurs pouvant prendre des valeurs différentes.

Etat du système

- État du système : l'ensemble des valeurs des indicateurs à l'instant t est appelé l'état du système à l'instant t.
- Effet d'une action : l'effet est caractérisé par 2 états du système :



t_0 = **état initial** de l'action A

t_1 = **état final** de l'action A

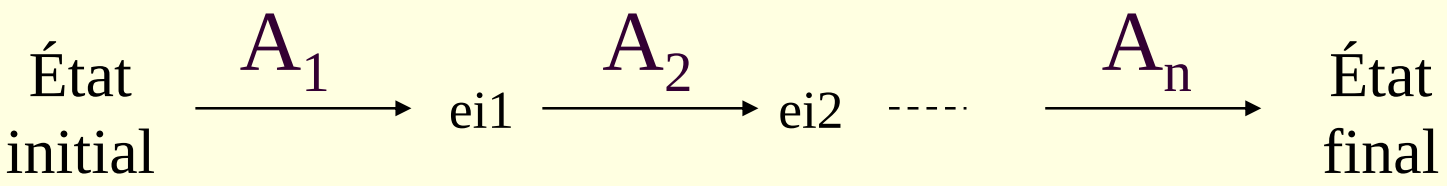
Algorithme

- **Spécification d'une action** : c'est la description des états initial et final de l'action
- **Algorithme** : suite logique d’actions permettant d’arriver à un résultat déterminé.

  Un algorithme décrit un comportement en termes d'actions réalisables a priori, et d'informations identifiées dans le lexique de l'algorithme (les indicateurs).

Construction d'un algorithme

- Construire un algorithme consiste à découvrir les actions qu'il faut organiser dans le temps, et à choisir la manière de les organiser pour obtenir le résultat escompté par leurs effets cumulés



Les indicateurs ou variables

- **Variable** : les valeurs des indicateurs changent (varient) sous l'effet des actions de l'algorithme. C'est pourquoi les indicateurs sont aussi appelés des variables
- Chaque information (nombre entier, caractère, …) que l'on veut utiliser doit être stockée dans un emplacement mémoire.

Exemples d'algorithmes

- Une recette de cuisine
- Un mode d'emploi
- Une stratégie gagnante pour un jeu simple
- Une méthode systématique pour résoudre un problème mathématique, par exemple la résolution d'un système d'équations
- Une notice de montage d’un meuble en kit.

Importance de l'algorithme

- Savoir expliquer comment faire un travail sans la moindre ambiguïté
  - langage simple : des instructions (opérations élémentaires)
  - suite finie d'actions à entreprendre en respectant une chronologie imposée
- L'écriture algorithmique : un travail de programmation à visée « universelle »

  un algorithme ne dépend a priori
  - ni du langage dans lequel il sera implanté,
  - ni de la machine qui exécutera le programme correspondant.

Étapes de construction d'un algorithme

1) Préciser les conditions de travail du problème :
   objets caractérisant les informations à manipuler
   répertoire d'actions dont on dispose
2) Réfléchir à une méthode bien définie pour résoudre le problème.
3) Décrire cette méthode sans ambiguïté

Étapes de construction d'un algorithme

4) Évaluer la méthode décrite (temps d'exécution, nombre d'opérations, taille, etc..)
5) Réfléchir aux qualités et aux défauts de la méthode choisie
6) Réfléchir à d'autres méthodes et les comparer selon les critères de qualité définis

Jouons un peu…

Trouver une solution pour gagner au « jeu des plaquettes » dont voici la règle :

- Il y a un meneur de jeu et 5 joueurs.
- Le meneur de jeu annonce un nombre entre 0 et 45.
- Les joueurs, doivent chacun choisir un nombre entre 0 et 9.
- Le total des chiffres choisis doit être égal au nombre annoncé par le meneur de jeu.
- **Les joueurs n'ont pas le droit de se concerter pendant le choix du nombre qu'ils vont annoncer**.

Jouons un peu…

- faire une partie pour comprendre le jeu (vous allez perdre !)
- réfléchir à une stratégie gagnante (vous ne devez plus perdre !)
- faire une partie pour vérifier que votre stratégie permet de gagner pour toutes les valeurs possibles

2 - Organisation séquentielle d’un calcul

2 - Organisation séquentielle d’un calcul

Exemple :

Écrire un algorithme qui, pour un nombre donné de secondes inférieur à un million, calcule son équivalent en nombre de jours, d’heures, de minutes et de secondes.

2.1 - Spécification, désignation, typage

Étapes de la construction d'un algorithme :

- Spécifier un problème consiste à expliciter les informations pertinentes pour une modélisation algorithmique, notamment les données et les résultats, puis à formuler les relations qui les caractérisent.
- Désigner les informations : c’est leur donner un nom

2.1 - Spécification, désignation, typage

Étapes de la construction d'un algorithme :

- Typer les informations, c’est donner leurs domaines de valeurs
- Formuler la solution algorithmique

Illustration sur l'exemple

Informations manipulées : le lexique de l'algorithme

Lexique

ns : entier entre 0 et 999999 // donnée : nombre de secondes
j : entier ≥ 0 // résultat : nombre de jours
h : entier entre 0 et 23 // résultat : nombre d'heures
m: entier entre 0 et 59 // résultat : nombre de minutes
s : entier entre 0 et 59 // résultat : nombre de secondes

Relation entre données et résultats :

ns = (86400 * j) + (3600 * h) + 60 * m + s

Illustration sur l'exemple

Algorithme

// ns = n j = ? h = ? m = ? s = ?

j ← ns div 86400 // div représente la division euclidienne

r1 ← ns mod 86400 // mod représente le modulo (reste de la division)

*// ns = n j = j_0 h = ? m = ? s = ? r1 = r_a n = j_0 * 86400 + r_a*

h ← r1 div 3600

r2 ← r1 mod 3600

// ns = n j = j_0 h = h_0 m = ? s = ? r1 = r_a r2 = r_b

*// n = j_0 * 86400 + h_0 * 3600 + r_b*

m ← r2 div 60

s ← r2 mod 60

// ns = n j = j_0 h = h_0 m = m_0 s = s_0 r1 = r_a r2 = r_b

*// n = j_0 * 86400 + h_0 * 3600 + m_0 * 60 + s_0*

Illustration sur l'exemple

lexique principal (complété)

ns : entier entre 0 et 999999 // donnée : nombre de secondes
j : entier ≥ 0 // résultat : nombre de jours
h : entier entre 0 et 23 // résultat : nombre d'heures
m: entier entre 0 et 59 // résultat : nombre de minutes
s : entier entre 0 et 59 // résultat : nombre de secondes

r1 : entier entre 0 et 86399 // intermédiaire : ns modulo 86400
r2 : entier entre 0 et 3599 // intermédiaire : r1 modulo 3600

2.2 - Lecture de données et écriture de résultats

On dispose de deux opérations :

- saisir: permet la saisie de données du clavier **cl**

  Forme : **cl**.saisir (liste des informations saisies)

  Exemple : cl.saisir(a,b)

- afficher : affichage des résultats sur l'écran **e**

  Forme : **e**.afficher (liste des valeurs affichées)

  Exemple : e.afficher("Total TTC: ",tt * 1,196)

2.2 - Lecture de données et écriture de résultats

On dispose de deux opérations :

- saisir: permet la saisie de données du clavier **cl**

  Forme : **cl**.saisir (liste des informations saisies)

  Exemple : cl.saisir(a,b)

  Définies dans le lexique (les données)

- afficher : affichage des résultats sur l'écran **e**

  Forme : **e**.afficher (liste des valeurs affichées)

  Exemple : e.afficher("Total TTC: ",tt * 1,196)

2.2 - Lecture de données et écriture de résultats

On dispose de deux opérations :

- saisir: saisie de données du clavier **cl**

  Forme : **cl**.saisir (liste des informations saisies)

  Exemple : cl.saisir(a,b)

- afficher: affichage des résultats sur l'écran **e**

  Forme : **e**.afficher (liste des valeurs affichées)

  - Noms d'informations (résultats calculés)
  - Constantes (chaînes de caractères par exemple)
  - Expressions

  Exemple : e.afficher("Total TTC: ",tt * 1.196)

Illustration sur l'exemple

lexique (ajout):
cl : clavier // périphérique d'entrée
e : écran // périphérique de sortie
Algorithme
// ns = ? j = ? h = ? m = ? s = ?
e.afficher("entrez un nombre de secondes inférieur à 1000000:")
cl.saisir (ns) *// ns = n*
j ← ns div 86400 ; r1 ← ns mod 86400
*// ns = n j = j_0 h = ? m = ? s = ? ns = j_0 * 86400 + r_a*
h ← r1 div 3600 ; r2 ← r1 mod 3600
*// ns = n j = j_0 h = h_0 m = ? s = ? ns = j_0 * 86400 + h_0 * 3600 + r_b*
m← r2 div 60 ; s ← r2 mod 60
// ns = n j = j_0 h = h_0 m = m_0 s = s_0
*// ns = j_0 * 86400 + h_0 * 3600 + m_0 * 60 + s_0*
e.afficher (ns," = ", j ," jours, ", h, " heures, ", m, " minutes, ", s, " secondes")

Types de base : **les entiers**

- Entiers : **Z**

  constantes : 156 -12 1998

  opérations : + - * / div mod

  $+ - * : Z \times Z \rightarrow Z$

  div mod : $Z \times Z^* \rightarrow Z$

  $/ : Z \times Z^* \rightarrow \mathbf{R}$

  comparaisons : $= \neq < > \leq \geq$

  $Z \times Z \rightarrow \mathbf{B} = \{ \text{vrai, faux} \}$

Types de base : **les réels**

- Réels : **R**

  constantes : 3,14159 -0.001 1.2E5

  opérations : + - * /

  + - * : **R** X **R** $\rightarrow$ **R**

  / : **R** X **R*** $\rightarrow$ **R**

  comparaisons : = $\neq$ < > $\leq$ $\geq$

  R X **R** $\rightarrow$ **B** = { vrai, faux }

Partie entière et partie décimale d'un réel

Fonctions prédéfinies : **pent** et **pdec**

pent : **R** $\rightarrow$ **Z**

$\forall$ x $\in$ **R**, pent(x) = partie entière de x

Exemple : pent(-34,12) = -34

pdec : **R** $\rightarrow$ [0,1[

$\forall$ x $\in$ **R**, pdec(x) = partie décimale de x

Exemple : pdec(-34,12) = 0,12

Exemples d'utilisation de pent et pdec

a, c : entiers

b, d : réels entre 0 et 1

a ← pent(12.56)

// a = 12

b ← pdec(-12.56)

// b = 0.56

c ← pent(12.56*100)

// c = 1256

// attention : 12.56 * 100 = 1256.0 (un réel et non un entier)

d ← pdec(12.56*100)

// d = 0.0

Types de base : **les booléens**

- Booléens : **B** = { vrai, faux }

  constantes : vrai, faux

  opérations : et ou (binaires) non (unaire)

  et ou : **B** X **B** $\rightarrow$ **B**

  non : **B** $\rightarrow$ **B**

  comparaisons : = $\neq$

  B X **B** $\rightarrow$ **B**

Rappel : Table de vérité

A	B	A et B	A ou B	non A	non B	non A et non B
V	V	V	V	F	F	F
V	F	F	V	F	V	F
F	V	F	V	V	F	F
F	F	F	F	V	V	V

non (A ou B) = non A et non B

non (A et B) = non A ou non B

Exemple d'utilisation de booléens

lexique principal
cl : clavier // périphérique d'entrée
e : écran // périphérique de sortie
a, b, c : entiers // données : valeurs à examiner
aEstMax : booléen // résultat : vrai si a plus grand que b et c
k : booléen // k est vrai si ?
Algorithme principal
e.afficher("entrez 3 entiers:")
cl.saisir (a, b, c) // $a = a_0$, $b = b_0$, $c = c_0$, *aEstMax = ?*
aEstMax ← (a > b) et (a > c) // *aEstMax = vrai si a > b et si a > c, faux sinon*
e.afficher(a, " plus grand que " , b, " et ", c, ':' , aEstMax)
k ← (b > c) ou aEstMax
e.afficher(k) // on affiche vrai si ?...

Types de base : **caractères**

- Ensemble des caractères : **C**

  constantes : 'A' 's' '=' ' ' (espace)

  opérations : néant !

  comparaisons : = ≠

$$\mathbf{C} \times \mathbf{C} \rightarrow \mathbf{B}$$

Attention aux notations !

'X' représente le caractère X majuscule

X est le nom d'une information (indicateur)

Si b est un booléen et c un caractère, que signifie cette instruction ?

$$b \leftarrow (c = \text{'A'})$$

Retour sur l'affectation d'une valeur à un indicateur (une variable)

Forme générale

NomIndicateur ← expression arithmétique ou logique

ou

NomIndicateur ← nom d'un autre indicateur

ou

NomIndicateur ← constante

• Instruction permettant d'attribuer à un indicateur identifié par le nom placé à gauche du symbole ← la valeur de l'élément placé à droite de ce symbole.

• La valeur à droite du symbole ← doit être **du même type** que celui de l'indicateur placé à gauche de ce symbole.

Exercice 1

- Trouvez le type de l'indicateur auquel on donne une valeur
- Trouvez les erreurs

```
valA ← 0.56
valB ← (a > b)
valC ← valC + 1
mot ← "Hello world !"
valD ← 'Z'
valE ← valA
valB ← ValC * 2
valF ← ValD + ValC
valD ← '1'
valD ← X
```

Exercice 2

Trouvez ce que fait l'algorithme suivant en décrivant les états intermédiaires

lexique

cl : clavier // périphérique d'entrée
a, b: entiers // données : valeurs à manipuler
c : entier // intermédiaire : pour quoi faire ?

Algorithme

cl.saisir(a)
cl.saisir(b) // $a = a_0$ $b = b_0$
$c \leftarrow a$ // $a = a_0$ $b = b_0$ $c = a_0$
$a \leftarrow b$ // $a = b_0$ $b = b_0$ $c = a_0$
$b \leftarrow c$ // $a = b_0$ $b = a_0$ $c = a_0$

Exercice 3

Trouvez ce que fait l'algorithme suivant en décrivant les états intermédiaires

lexique

cl : clavier // périphérique d'entrée

a, b: entiers // données : valeurs à manipuler

Algorithme

cl.saisir(a)

cl.saisir(b)

a ← a + b

b ← a – b

a ← a – b

Affectation : complément

- Il est possible d'écrire plusieurs instructions sur une même ligne. Dans ce cas on sépare les instructions par un **;**
- Exemples :

  cl.saisir(a) ; e.ecrire(a)

  i ← i + 1 ; k ← 1

Types de base : **chaines de caractères**

- Chaînes de caractères : **C***

  constantes : "A" "Bon" ""

  opération de concaténation :

  & : **C*** X **C*** $\rightarrow$ **C*** (concaténation)

"Bon" & "jour" a pour résultat: "Bonjour"

"" & "hello" a pour résultat: "hello"

si m1, m2 et m3 sont des chaines, quel est l'effet de cette séquence d'instructions ?

m1 $\leftarrow$ "soir" ;

m2 $\leftarrow$ "as"

m3 $\leftarrow$ m2 & m1

Types de base : **chaines de caractères**

comparaisons de chaines : $=$ $\neq$

$$\mathbf{C^*} \times \mathbf{C^*} \rightarrow \mathbf{B}$$

Attention aux notations !

"X" représente la chaîne de caractères comprenant un X majuscule

X est le nom d'une information

'X' représente le caractère X majuscule

Fonctions et opérations prédéfinies sur les chaînes : **concaténation, ajout d'un caractère**

- concaténation : **&**

  CH & "jour"

- ajout d'un caractère à gauche d'une chaîne: ∘

  ∘ **C** X **C*** $\rightarrow$ **C$^+$** (C^+ représente les chaînes non vides)

  'c' ∘ "abcd" = "cabcd"

  'S' ∘ "" = "S"

- ajout d'un caractère à droite d'une chaîne: •

  • **C*** X **C** $\rightarrow$ **C$^+$** (C^+ représente les chaînes non vides)

  "abcd" • 'c' = "abcdc"

Exercice

- Parmi les expressions suivantes, lesquelles sont incorrectes ?
  - 123+A
  - ‘A’ • ‘B’
  - ("A" • ‘B’) & ‘c’
  - ("A" • ‘B’) & "c"
  - "abc" & ‘def’
  - "abc" • ‘def’
  - "abc" & "" & "def"
  - ‘’ ∘ "abc"
  - ‘ ’ ∘ "abc"
  - 23 • ‘A’

Exercice

- Parmi les expressions suivantes, lesquelles sont incorrectes ?
  - 123+A
  - ‘A’ • ‘B’ • s’applique entre une chaine et un caractère
  - ("A" • ‘B’) & ‘c’ & s’applique entre deux chaines
  - ("A" • ‘B’) & "c"
  - "abc" & ‘def’ 'def' ne correspond à rien :
  - "abc" • ‘def’ ce n’est ni un caractère ni un chaine
  - "abc" & "" & "def"
  - ‘’ ∘ "abc" aucun caractère entre les ‘ : ce n’est pas un car.
  - ‘ ’ ∘ "abc"
  - 23 • ‘A’ 23 est un entier

Fonctions et opérations prédéfinies sur les chaînes : **sous-chaine, longueur**

- Fonctions prédéfinies
  - **SousChaine : C* x N* x N → C***

    SousChaine(ch,d,long) désigne la sous-chaîne de ch de longueur long et commençant avec le d^{ième} caractère

    SousChaine("hello",2,3)= "ell"

    SousChaine("hello",3,10)= "llo"
  - **Longueur : C* → N**

    Longueur(ch) désigne la longueur de la chaîne ch

    c'est-à-dire le nombre de caractères composant ch

    Longueur("hello") = 5

    Longueur("") = 0

    Longueur(SousChaine("hello",3,10)) = 3

Fonctions et opérations prédéfinies sur les chaînes : **sélection d'un caractère dans une chaine non vide**

Sélection d'un caractère particulier d'une chaine **non vide**:

nième : $\mathbf{C^+}$ X **N*** $\rightarrow$ **C**

Soit **ch** une chaîne non vide

nième(ch,1) désigne le 1er caractère de ch

nième(ch, i) désigne le ième caractère de ch, **1 ≤ i ≤ longueur(ch)**

nième(ch, longueur(ch)) désigne le dernier caractère de la chaîne ch

Exemple : *// ch = "hello"*

nième(ch,2) = 'e'

Exercice

- Ecrire un algorithme qui saisit un verbe du premier groupe et affiche sa conjugaison au présent de l'indicatif.

Conjugaison d'un verbe du premier groupe au présent de l'indicatif

lexique
cl : clavier // périphérique d'entrée
e : écran // périphérique de sortie
verbe : chaine // donnée : verbe à conjuguer
radical : chaine // intermédiaire : radical du verbe à conjuguer
Algorithme
e.afficher("entrez un verbe du premier groupe:")
cl.saisir (verbe) *// verbe = v_0*
radical ← sousChaine(verbe, 1, longueur(verbe) - 2) *// radical = radical de v_0*
e.afficher ("je ", radical, 'e')
e.afficher ("tu ", radical & "es")
e.afficher ("il ou elle ", radical • 'e')
e.afficher ("nous ", radical, "ons")
e.afficher ("vous ", (radical • 'e') • 'z')
e.afficher ("ils ou elles ", (radical • 'e') & "nt")

Constantes nommées

Dans le lexique d'un algorithme, il est possible de définir des indicateurs dont la valeur est fixée pour tout l'algorithme. Leur valeur n'est pas modifiable.

constante nom le type valeur

Exemples :

constante PI le réel 3.14159265

constante MAX l'entier 100

constante DOUBLEMAX l'entier MAX * 2

constante MESSAGE la chaine "Hello world !"

Conventions d’écriture

- Les constantes peuvent être nommées par des noms formés uniquement de majuscules (standard en C et C++)
- Les autres variables par des noms commençant par une minuscule (standard Java)
- Les noms de types et de classe peuvent êtres des noms commençant par une majuscule (standard Java)
- Il n’y a pas de règle absolue, mais des habitudes ou des standards à respecter
- Il faut être cohérent en suivant toujours les mêmes principes